AMENDMENTS TO THE CLAIMS

Please amend the claims as follows.

1. – 25. (Cancelled)

26. (Currently Amended) A method for handling sharing of a physical memory space between a first process and a second process, the method comprising:

allocating a portion of the physical memory space and creating a buffer object responsive to a buffer object request, wherein one of the processes executes native code of a program and the other process executes safe language code of the program; [[and]]

mapping a first address range of the first process and a second address range of the second process to the allocated portion of the shared physical memory space, wherein the buffer object represents at least one of the address ranges, wherein address ranges in the first process to overlap and/or nest;

requesting, by the first process, an address range [A2, A2+S2] that overlaps with a previously allocated address range [A1, A1+S1], wherein A1 and A2 represent addresses in the first process and S1 and S2 represent memory space sizes;

the second process,

allocating an address range [A2', A2'+S2], wherein A2' represent addresses in the second process,

mapping [A2', A2' + (A1+S1-A2)] in the second process to a same first portion of the one or more physical memory spaces to which [A2, A1+S1] in the first process is mapped, and

mapping [A2'+(A1+S1-A2+S2] in the second process to a same second portion of the one or more physical memory spaces to which [A1+S1, A2+S2] in the first process is mapped, wherein $A1 < A2 < A1+S1 < A2+S2$.

27. (Previously presented) The method of claim 26, wherein at least one of the address ranges comprises virtual addresses.

28. (Previously presented) The method of claim 26, wherein the physical memory space comprises a set of one or more physical pages.

29. (Previously presented) The method of claim 26, further comprising the second process creating the buffer object and indicating the allocated physical memory space portion and the buffer object to the first process.

30. (Previously presented) The method of claim 29, wherein the safe language code comprises Java language code.

31. (Previously presented) The method of claim 29 further comprising the first process indicating the buffer object to one or more callers.

32. (Previously presented) The method of claim 29 further comprising maintaining an encoding that indicates allocated portions of the physical memory space to allow detection of at least one of overlapping address ranges and nested address ranges.

33. – 35. (Cancelled)

36. (Previously presented) The method of claim 50, further comprising communicating an identifier of the created direct buffer to one or more callers.

37. (Previously presented) The method of claim 50, wherein the memory space comprises a set of one or more physical memory pages.

38. (Cancelled)

39. (Cancelled)

40. (Previously presented) The method of claim 50 further comprising maintaining a list that indicates mapped address ranges to allow detection of at least one of overlapping address ranges and nested address ranges.

41. (Previously presented) The method of claim 50, wherein at least one of the first process address range and the second process address range comprise virtual addresses.

42. (Cancelled)

43. (Cancelled)

44. (Cancelled)

45. (Cancelled)

46. (Cancelled)

47. (Currently amended) An apparatus comprising:

a memory; and

means for mapping an address range in a first process for a first code and an address range in a second process for a second code to a same region of the memory to facilitate transparent code isolation, and for representing at least one of the address ranges with a buffer object, wherein address ranges in the first process to overlap and/or nest;

means for requesting, by the first process, an address range [A2, A2+S2] that overlaps with a previously allocated address range [A1, A1+S1], wherein A1 and A2 represent addresses in the first process and S1 and S2 represent memory space sizes;

means for allocating, by the second process, an address range [A2', A2'+S2], wherein A2' represent addresses in the second process,

means for mapping, by the second process, [A2', A2' + (A1+S1-A2)] in the second process to a same first portion of the one or more physical memory spaces to which [A2, A1+S1] in the first process is mapped, and

means for mapping, by the second process, [A2'+(A1+S1-A2+S2] in the second process to a same second portion of the one or more physical memory spaces to which [A1+S1, A2+S2] in the first process is mapped, wherein A1 < A2 < A1+S1 < A2+S2, and

wherein one of the first and second processes executes native code and the other of the first and second processes executes safe language code.

48. (Previously presented) The apparatus of claim 47 further comprising means for communicating an identifier of the buffer object between the first and second processes.

49. (Previously presented) The apparatus of claim 47 further comprising means for detecting at least one of nested address ranges and overlapping address ranges.

50. (Currently Amended) A method of performing native code isolation in the presence of direct buffers without losing transparency, the method comprising:

mapping a first address range of a the first process and a second address range of a second process to a same portion of memory space, wherein one of the first and second processes executes native code and the other of the first and second processes executes a safe language code; [[and]]

creating a direct buffer to represent at least one of the first and the second address ranges, wherein address ranges in the first process to overlap and/or nest;

requesting, by the first process, an address range [A2, A2+S2] that overlaps with a previously allocated address range [A1, A1+S1], wherein A1 and A2 represent addresses in the first process and S1 and S2 represent memory space sizes;

the second process,

allocating an address range [A2', A2'+S2], wherein A2' represent addresses in the second process,

mapping [A2', A2' + (A1+S1-A2)] in the second process to a same first portion of the one or more physical memory spaces to which [A2, A1+S1] in the first process is mapped, and

mapping [A2'+(A1+S1-A2+S2] in the second process to a same second portion of the one or more physical memory spaces to which [A1+S1, A2+S2] in the first process is mapped, wherein $A1 < A2 < A1+S1 < A2+S2$.

51. (Previously presented) The method of claim 50, wherein the direct buffer is created by the second process responsive to receiving a direct buffer request from the first process.

52. (Previously presented) The method of claim 50, wherein the same portion of memory space is allocated from one of a memory mapped file and a frame buffer.

53. (Cancelled)